课程回顾

- 贪心算法
  - 贪心算法原理
    - 一般设计步骤
    - 贪心算法要素：贪心选择性质、最优子结构
  - 0-1背包 vs. 分数背包
  - 贪心算法理论：拟阵
    - 加权拟阵中寻找最大权重的独立子集问题->贪心算法可求得最优解
  - 贪心算法应用：最优装载、单源最短路径、最小生成树
- 近似算法
  - 近似比

近似算法概述 (续)

■最优化问题Π（最小化问题/最大化问题）：

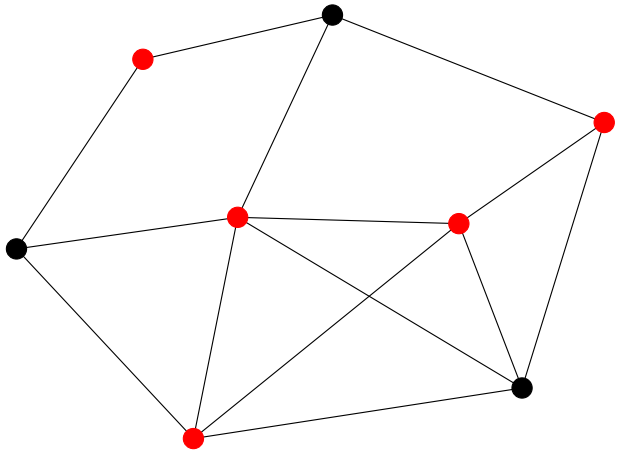
- 近似方案/近似模式（approximation scheme）：
  输入为(I, ε)，其中$I \in D$，$\varepsilon>0$为误差参数，
  若有$(1-\varepsilon)\mathrm{OPT}(I) \le \mathrm{SOL}_A(I, \varepsilon) \le (1+\varepsilon)\mathrm{OPT}(I)$，则称算法$A$是NP-Hard最优化问题Π的近似方案/近似模式
- PTAS (Polynomial-Time Approximation Scheme)：对于每一个确定的$\varepsilon>0$，该近似方案的运行时间以I的规模的多项式为上界
- FPTAS (Fully Polynomial-Time Approximation Scheme)：对于每一个确定的$\varepsilon>0$，该近似方案的运行时间以I的规模和$1/\varepsilon$的多项式为上界

FPTAS被认为是最值得研究的近似算法，仅有极少的NP-Hard问题存在FPTAS

顶点覆盖判定问题

■顶点覆盖问题VERTEX-COVER

- 给定一个无向图$G=(V, E)$和一个正整数k，判定是否存在$V' \subseteq V$和$|V'|=k$，使得对任意$(u,v) \in E$有$u \in V'$或$v \in V'$，如果存在，就称V'为图G的一个大小为k的顶点覆盖
- 即E中每条边至少有一个顶点在V'中



存在一个规模k为5的顶点覆盖

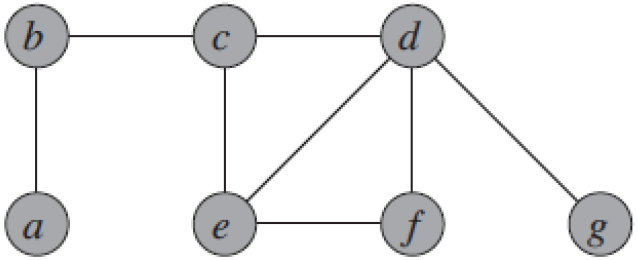
顶点覆盖问题

- 顶点覆盖的规模：V'包含的顶点数
- 顶点覆盖问题：在一个给定的无向图中找出一个具有最小规模的顶点覆盖（最优顶点覆盖）

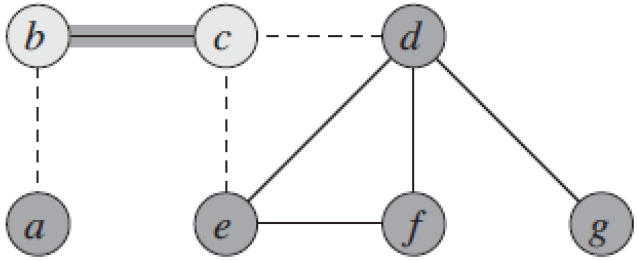
```
APPROX_VERTEX_COVER(G)
1   C ← ∅
2   E' ← G.E
3   while E' ≠ ∅ do
4       选择E'中任一条边(u, v)
5       C ← C ∪ {u, v}
6       将E'中与u及v邻接的边全部删除
7   return C
```

$O(|V|+|E|)$

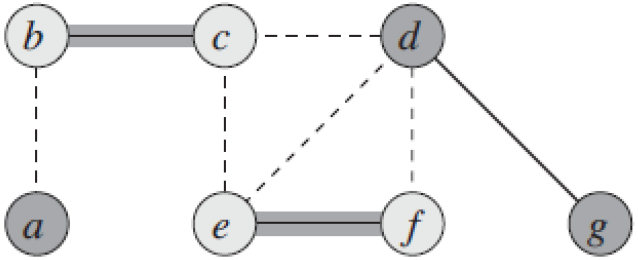
顶点覆盖问题 (续)



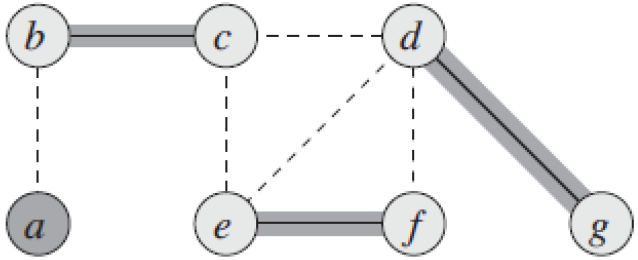
(a)



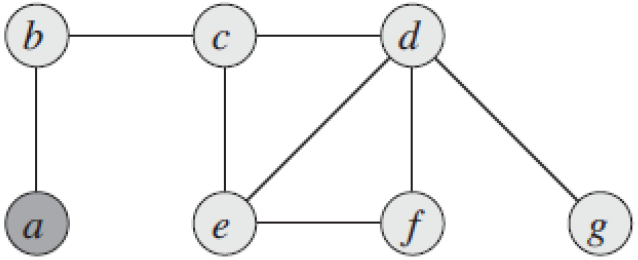
(b)



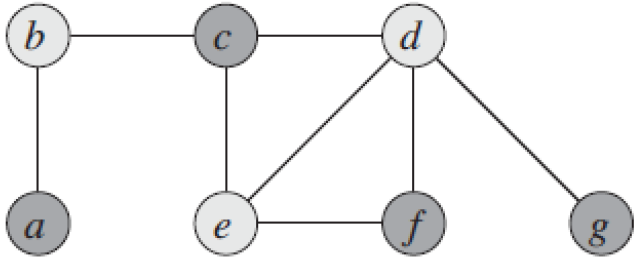
(c)



(d)



(e)



(f)

顶点覆盖问题 (续)

■APPROX_VERTEX_COVER算法近似比为2，即返回的规模最多是最优顶点覆盖规模的2倍

- 证明：（通过算法第4行选出的边的数量进行过渡）

  设算法第4行选出的边的数量为$|A|$，则$|C| = 2|A|$；

  设最优解为C^*，因为A中的边不共用顶点，则A中每条边至少有一个顶点要在C^*中，即 $|C^*| \geq |A|$；

  综合得到$|C| \leq 2|C^*|$，即 $\mathrm{SOL/OPT} \leq 2$

```
APPROX_VERTEX_COVER(G)
1   C ← ∅
2   E' ← G.E
3   while E' ≠ ∅ do
4       选择E'中任一条边(u, v)
5       C ← C ∪ {u, v}
6       将E'中与u及v邻接的边全部删除
7   return C
```

旅行商问题TSP

■旅行商问题TSP (Traveling Salesman Problem)

- 给定一个无向完全图$G=(V, E)$及定义在$V\times V$上的一个费用函数c和一个整数k，判定G是否存在经过V中各顶点恰好一次的回路，使得该回路的费用不超过k
- 一个售货员必须访问n个城市，售货员希望恰好访问每个城市一次，并最终回到出发城市。售货员从城市i到城市j的旅行费用为一个整数$c(i, j)$，旅行所需的全部费用是他旅行经过的各边费用之和，售货员希望整个旅行费用最低。判定问题为：旅行费用不超过k。

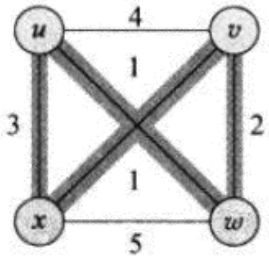


图 34-18 旅行商问题的一个实例。阴影覆盖的边表示费用最低的旅行路线，其费用为 7

旅行商问题TSP (续)

- 输入：无向完全图 $G=(V, E)$，每条边$(u, v)\in E$有一个非负整数代价$c(u, v)$
- 输出：G的一条具有最小代价的哈密顿回路（G中每个顶点只经过一次）
- 实际情况中，代价通常满足三角不等式，即：
  $c(u, w) \leq c(u, v) + c(v, w)$
  即：直达代价更小

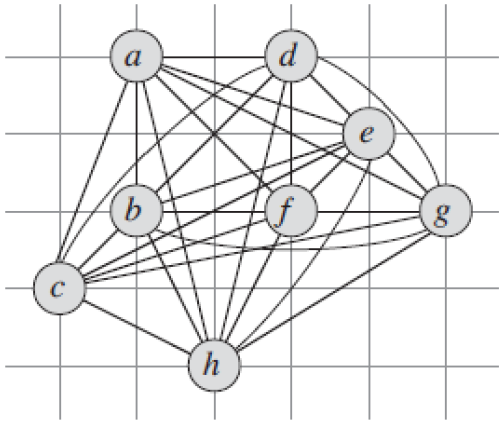
旅行商问题TSP (续)

■满足三角不等式的TSP：最小生成树法求近似解

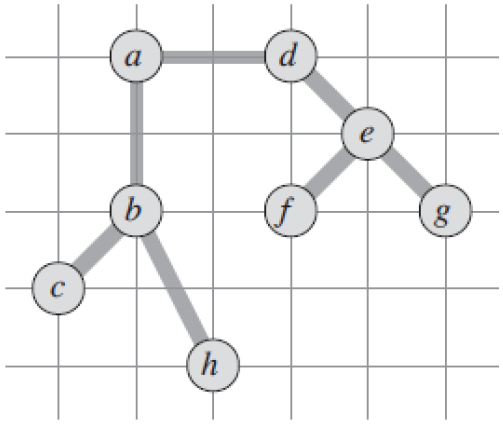
APPROX_TSP_TOUR(G, c)
1 选择一个顶点$r \in G.V$
2 使用Prim算法构造图G以r为根结点的最小生成树T
3 先序遍历T，生成顶点序列H
4 **return** 依次经过序列H中顶点、并最后回到第一个顶点的路径

- 运行时间：$\Theta(|V|^2)$
- 算法近似比：2
  - 证明：最小生成树的权值是最优旅行路线的下界：$c(T) \leq OPT$
    设仅能在最小生成树的边上进行旅行，那么先序遍历并回到根结点的代价：$c(W)=2c(T)$
    由于三角不等式，算法输出的代价$SOL \leq c(W)$
    由此得到：$SOL \leq 2OPT$，即$SOL/OPT \leq 2$

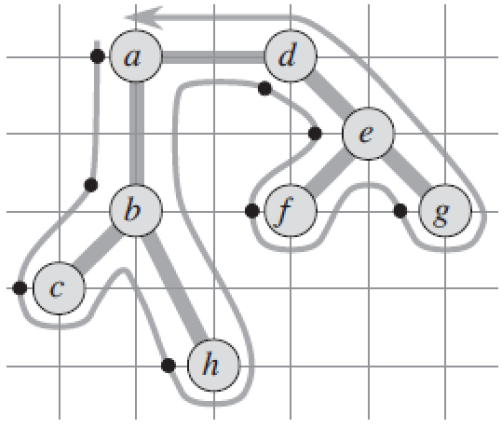
旅行商问题TSP (续)



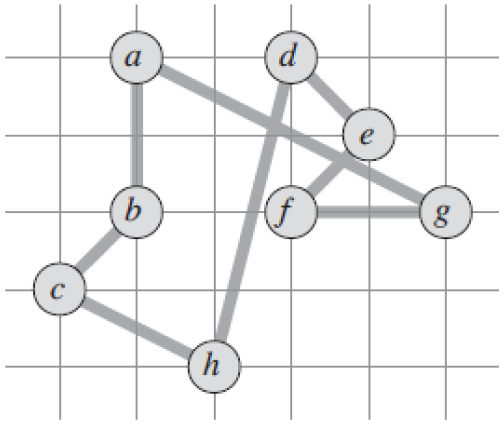
(a)



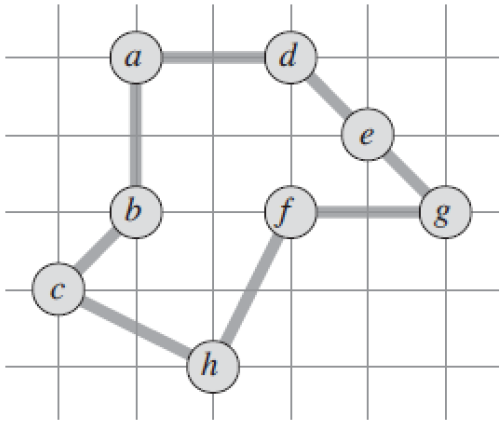
(b)



(c)



(d)

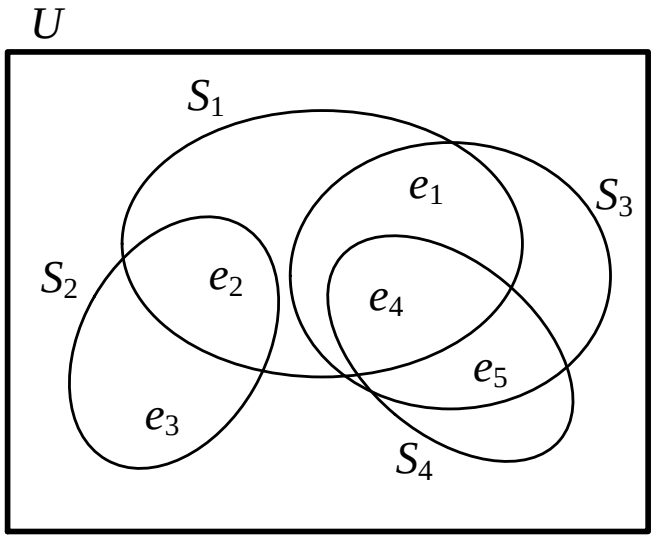


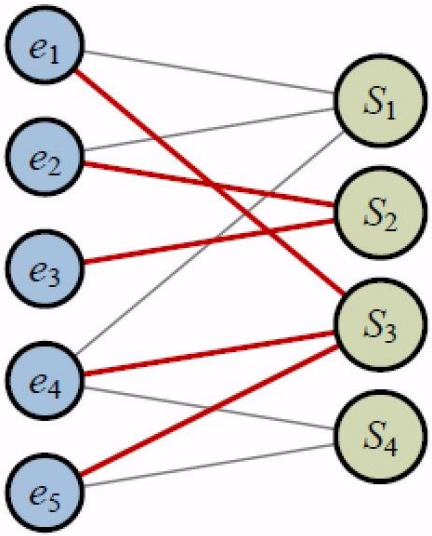
(e)

旅行商问题TSP (续)

- 一般TSP：P≠NP情况下，不存在多项式时间内具有常数近似比的近似算法（定理35.3）

集合覆盖问题





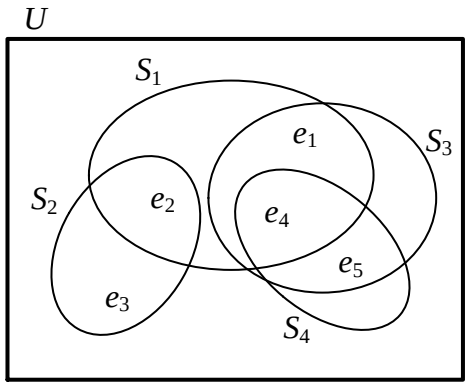
Set Cover

Given a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, ..., S_m\}$, and a cost function $c : \mathcal{S} \to \mathbf{Q}^+$, find a minimum cost subcollection of $\mathcal{S}$ that covers all elements of U.

集合覆盖问题 (续)

- 代价函数通常为解集合族中集合的数量，即用最少的集合覆盖所有的元素
- 贪心近似算法：每次选择覆盖元素最多的集合

```
GREEDY_SET_COVER(U, S)
1   X ← U
2   F ← ∅
3   while X ≠ ∅ do
4       选择使得|Si∩X|最大的Si∈S
5       X ← X - Si
6       F ← F ∪ {Si}
7   return F
```



多项式时间的
(ln|U|+1)近似算法

子集和问题

与0-1背包问题联系：
物品——正整数
物品价值——正整数的值
物品重量——正整数的值
背包容量——整数和t

■子集和问题SUBSET-SUM

- 给定一个正整数有限集S和一个整数目标$t>0$，判定是否存在S的一个子集$S' \subseteq S$，使得S'中的整数的和为t
- 例：S={1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993}, t =138457，则子集S'={1, 2, 7, 98, 343, 686, 2409, 17206, 117705}是该问题的一个解
- 最优化问题：找到子集$S' \subseteq S$，使得S'中的整数的和尽可能大，但不能超过t
- 存在$O(|S|t)$的伪多项式时间算法

子集和问题 (续)

■FPTAS算法：

以下设$n = |S|$，且$S=\{x_1, x_2, ..., x_n\}$

对任意$\varepsilon>0$，令$k = \left\lfloor \frac{\varepsilon x_{\max}}{n} \right\rfloor$，其中$x_{\max} = \max\limits_{1\le i\le n} x_i$

设置$x_i' = \lfloor x_i/k \rfloor, \quad i = 1, 2, ..., n$

返回以x_i'为物品价值、x_i为物品重量、t为背包容量的0-1背包动态规划算法的解

- 时间复杂度：$O(n^3/\varepsilon)$
- 近似比：SOL/OPT $\ge$ 1-ε

本章小结

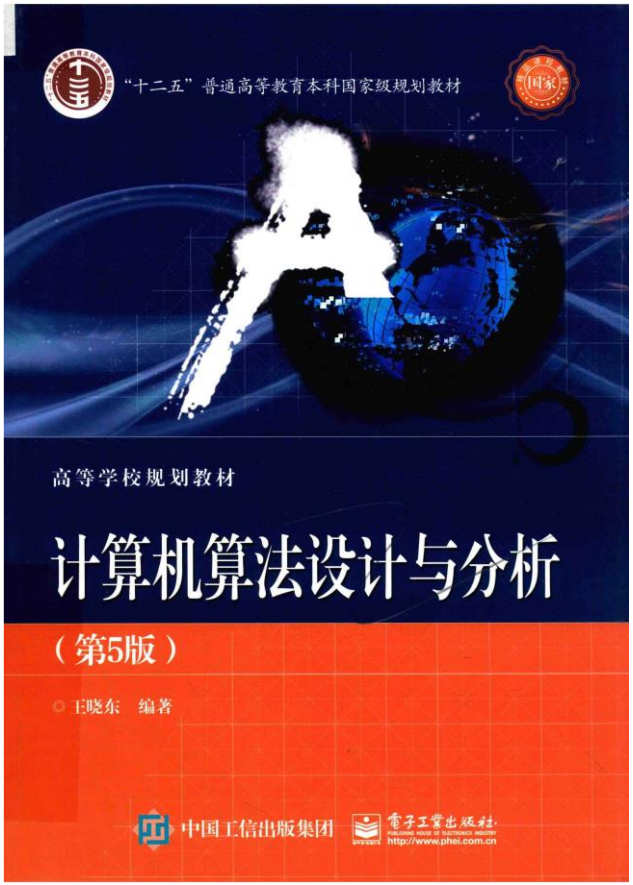
- 近似算法：用于解决NP-Hard问题的方案
- 近似比
- 近似算法举例：顶点覆盖问题、旅行商问题、集合覆盖问题、子集和问题

第6章 回溯法

苏州大学 计算机科学与技术学院
汪笑宇
Email: xywang21@suda.edu.cn

参考书目

■接下来几章内容可参考以下书本：



王晓东 编著
《计算机算法设计与分析》(第5版)
电子工业出版社

本章内容

- 回溯法算法框架：问题解空间，基本思想，递归回溯，迭代回溯，子集树与排列树
- 实际问题：装载问题，批处理作业调度，符号三角形问题，n后问题，0-1背包问题，最大团问题，图的m着色问题，TSP，圆排列问题，电路板排列问题，连续邮资问题
- 回溯法效率分析

回溯法概述

- 回溯法有“通用的解题法”之称，可以系统地搜索一个问题的所有解或任一解
- 在问题解空间树中按照深度优先策略，从根结点出发搜索解空间树
- 适合解组合数较大的问题

回溯法算法框架

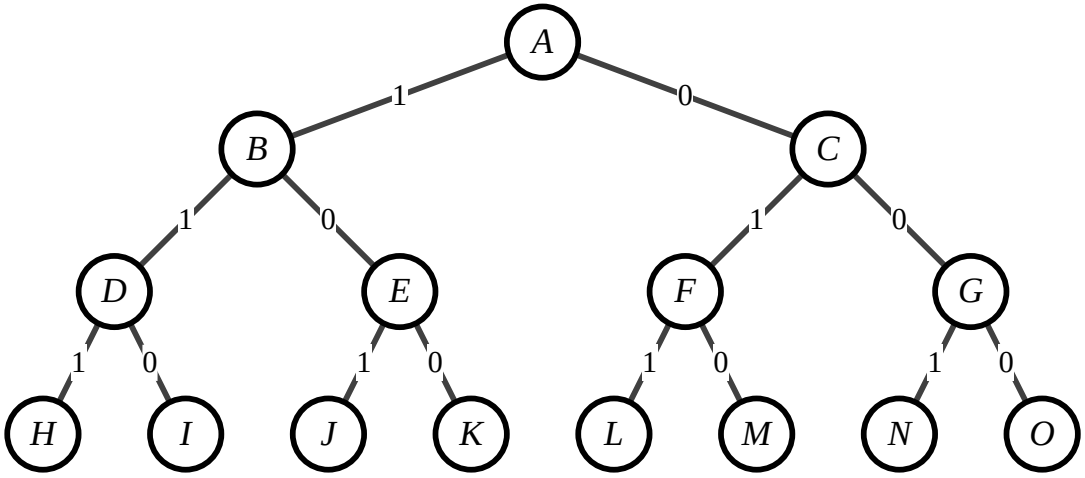
■问题的解空间

➢应明确定义，至少包含问题的一个（最优）解

➢例：0-1背包问题

解空间：长度为n的0-1向量，第i位为1表示选取物品i

如n=3时，解空间为{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)}

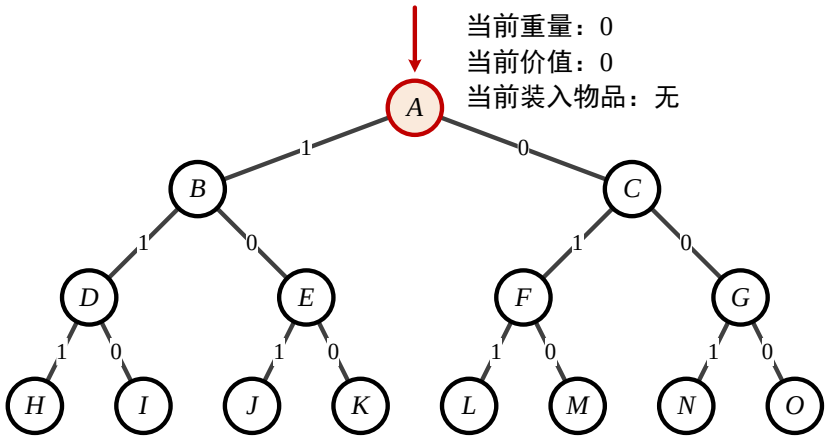


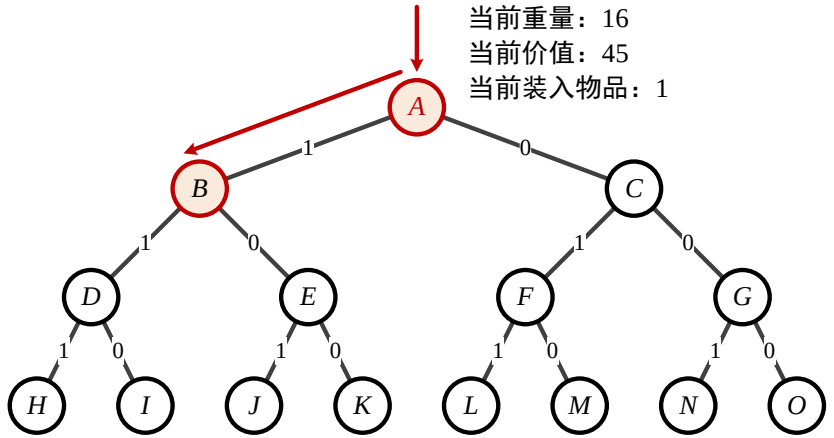
回溯法算法框架 (续)

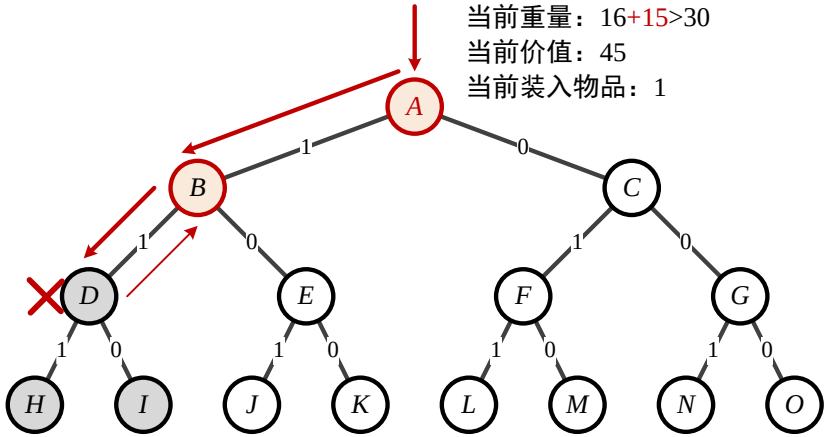
- 回溯法基本思想
  - 从开始结点（根结点）出发，以深度优先方式搜索整个解空间
  - 当前结点为扩展结点，可继续纵深搜索的结点为活结点，无法纵深搜索的为死结点
  - 扩展结点为死结点时，回溯至最近的一个活结点处，这个活结点作为扩展结点，递归求解，直至找到解或无活结点为止

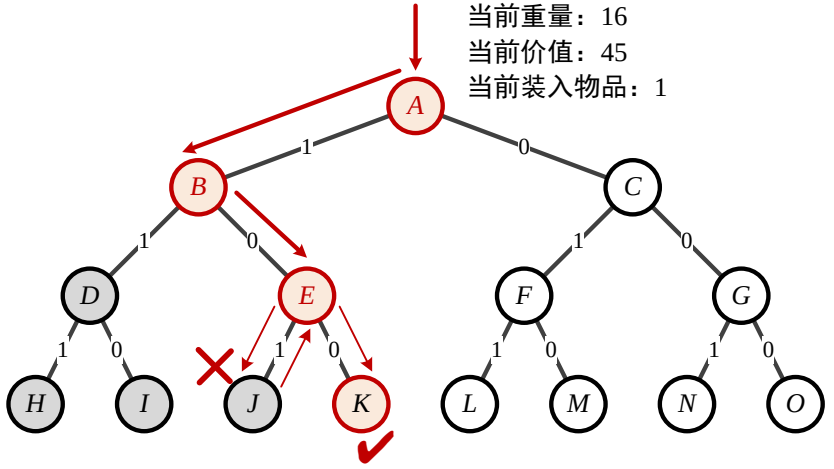
回溯法算法框架 (续)

- 例1：0-1背包问题：w=[16, 15, 15], v=[45, 25, 25], W=30



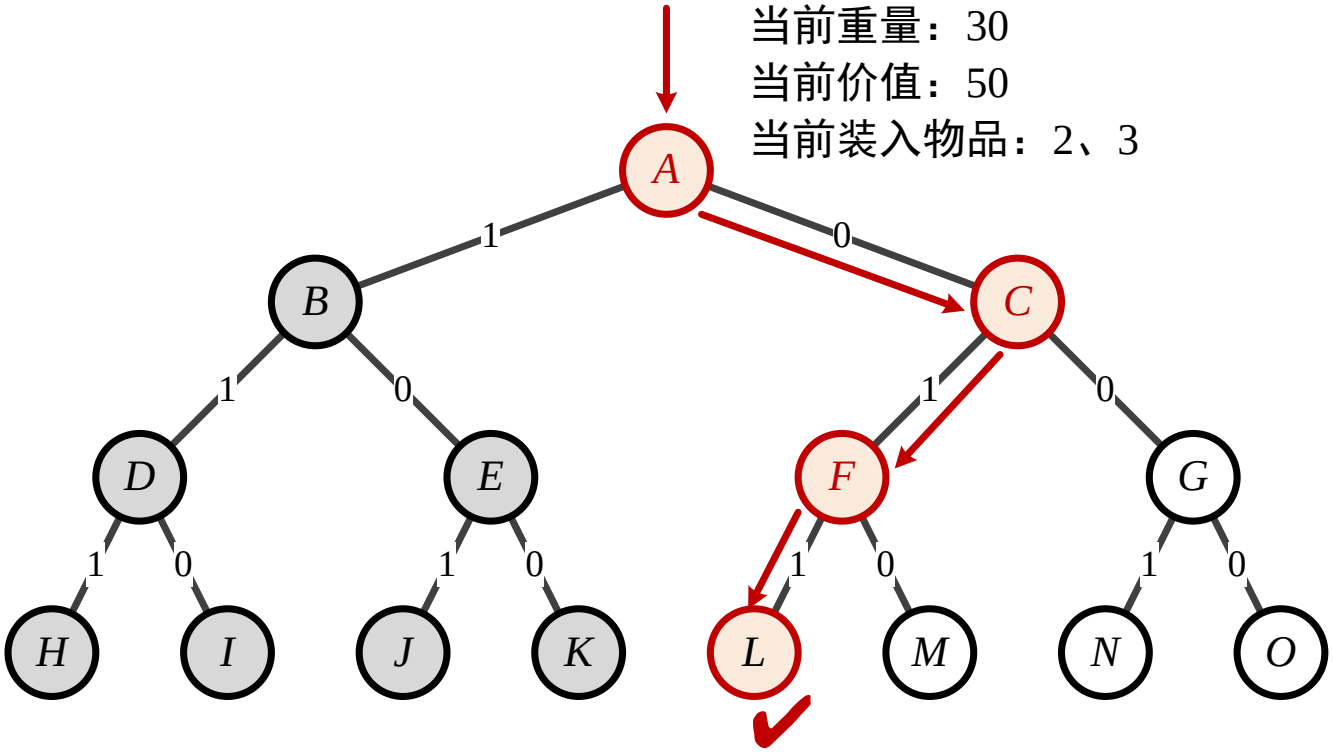






回溯法算法框架 (续)

- 例1：0-1背包问题：w=[16, 15, 15], v=[45, 25, 25], W=30



回溯法算法框架 (续)

- 例2：旅行售货员问题

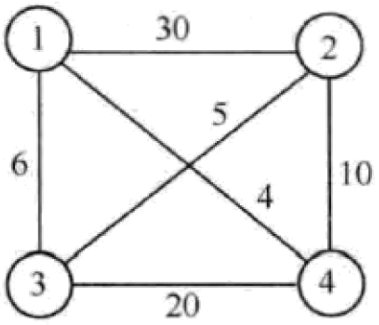


图 5-2　4 顶点带权图

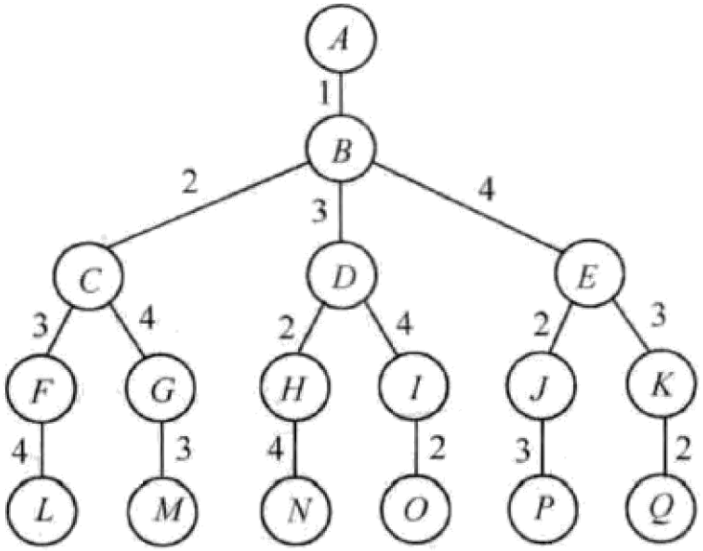


图 5-3　旅行售货员问题的解空间树

回溯法算法框架 (续)

■回溯法基本思想

➢采用剪枝函数避免无效搜索

- 约束函数在扩展结点处剪去不满足约束的子树
- 限界函数剪去得不到最优解的子树

■回溯法步骤：

1. 针对所给问题，定义问题的解空间
2. 确定易于搜索的解空间结构
3. 以深度优先方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索

回溯法算法框架 (续)

- 用回溯法解题的一个显著特征是在搜索过程中动态产生问题的解空间
  - 在任何时刻，算法只保存从根结点到当前扩展结点的路径
  - 如果解空间树中从根结点到叶结点的最长路径的长度为$h(n)$，则回溯法所需的计算空间通常为$O(h(n))$
  - 显式地存储整个解空间则需要$O(2^{h(n)})$或$O(h(n)!)$内存空间

回溯法算法框架 (续)

■递归回溯

```
BACKTRACK(x, t, n)  //n为总递归深度
1   if t > n
2       OUTPUT(x)   //x为可行解
3   elseif f(n, t) > g(n, t)  //当前无可搜索子树
4       return
5   else for i ← f(n, t) to g(n, t) do
6       x[t] ← h(i)
7       if CONSTRAINT(t) and BOUND(t)
8           BACKTRACK(x, t+1, n)
```

尾递归

- $f(n, t)$, $g(n, t)$：当前扩展结点处未搜索过的子树的起始编号和终止编号
- $h(i)$：当前扩展结点处$x[t]$的第i个可选值
- CONSTRAINT(t), BOUND(t)：当前扩展结点处的约束函数和限界函数

回溯法算法框架 (续)

■迭代回溯

```
ITERATIVE_BACKTRACK(n)
1   t ← 1
2   let x[1..n] be a new array
3   while t ≤ n do
4       if f(n, t) ≤ g(n, t)
5           for i ← f(n, t) to g(n, t) do
6               x[t] ← h(i)
7               if CONSTRAINT(t) and BOUND(t)
8                   t ← t + 1
9                   break
10      else t ← t - 1
11  OUTPUT(x)
```

回溯法算法框架 (续)

■子集树与排列树

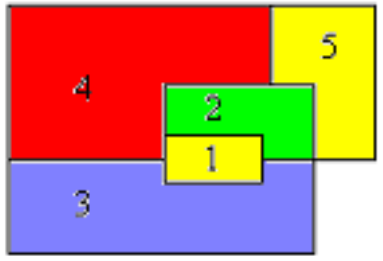
- 当所给的问题是从n个元素的集合S中找出满足某种性质的子集时，相应的解空间树称为子集树
  - 结点数：2^n
  - 遍历时间：$\Omega(2^n)$
- 当所给的问题是确定n个元素满足某种性质的排列时，相应的解空间树称为排列树
  - 结点数：$n!$
  - 遍历时间：$\Omega(n!)$

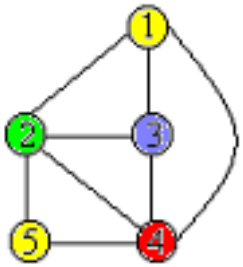
回溯法求解举例——0-1背包问题

- 解空间：子集树
- 基本思想：在搜索解空间树时，只要其左儿子结点是一个可行结点，搜索就进入其左子树。当右子树中有可能包含最优解时才进入右子树搜索，否则将右子树剪去
- 可行性约束函数：背包容量限制$\sum_{k=1}^{n} w_k x_k \leq W$
- 限界函数：剩余容量装入剩余物品的分数背包问题的最大价值
- BOUND：$O(n)$，最坏情况$O(2^n)$个结点都要计算BOUND函数
- 时间复杂度：$O(n2^n)$

回溯法求解举例——图的m着色问题

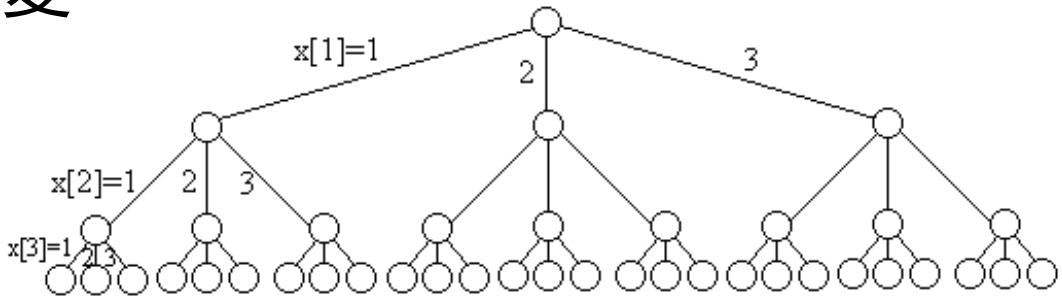
- 判定问题：给定无向连通图G和m种不同的颜色，用这些颜色为图G的各顶点着色，每个顶点着一种颜色，判断是否有一种着色法使G中每条边的2个顶点着不同颜色
- 最优化问题：若一个图最少需要m种颜色才能使图中每条边连接的2个顶点着不同颜色，则称这个数m为该图的色数。求一个图的色数m的问题称为图的m可着色优化问题
- 四色猜想





回溯法求解举例——图的m着色问题 (续)

- 解向量$x[1..n]$： $x[i]$表示顶点i所着颜色序号
- 可行性约束函数：顶点i与已着色的相邻顶点颜色不重复



- 时间复杂度：
  - 内结点个数：$\sum_{i=0}^{n-1} m^i$
  - 每个内结点检查子结点颜色可用性：$O(mn)$
  - 总耗时：$O(mn\sum_{i=0}^{n-1} m^i) = O(nm^n)$

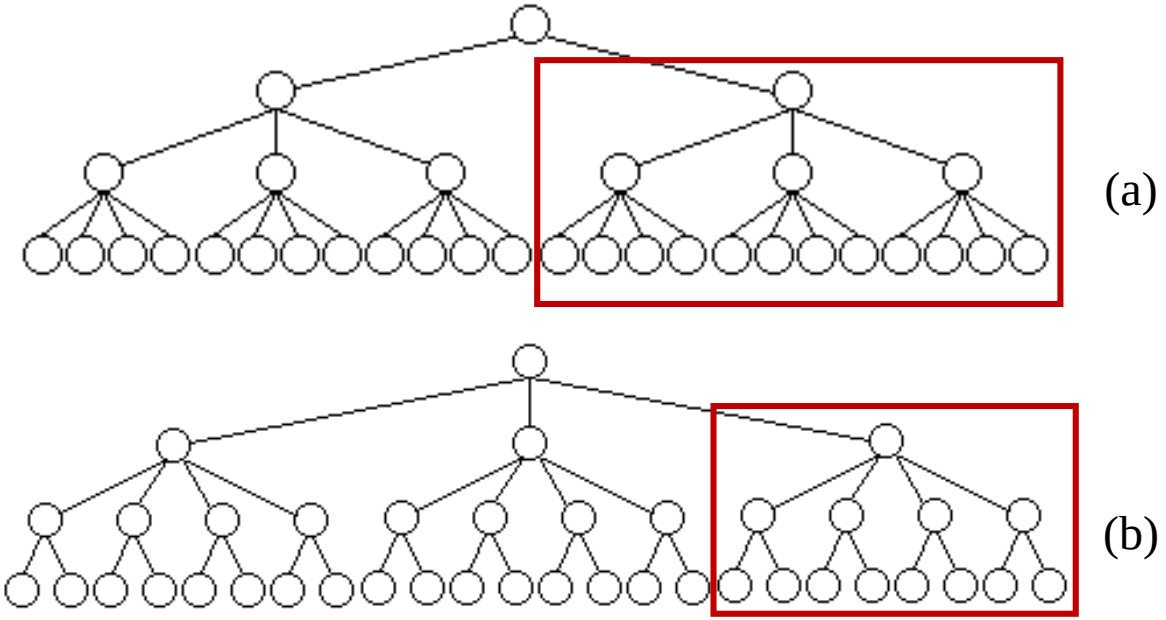
回溯法效率分析

- 回溯法效率依赖于：
  - 产生$x[k]$的时间
  - 满足显约束的$x[k]$值的个数
  - 计算约束函数CONSTRAINT的时间
  - 计算限界函数BOUND的时间
  - 满足约束函数和限界函数约束的所有$x[k]$的个数
- 好的约束函数能显著减少所生成的结点数，但这样的约束函数往往计算量较大

回溯法效率分析 (续)

■重排原理

➢对于许多问题而言，在搜索试探时选取$x[i]$的值顺序是任意的。在其它条件相当的前提下，让可取值最少的$x[i]$优先



第7章 分支限界法

苏州大学 计算机科学与技术学院
汪笑宇
Email: xywang21@suda.edu.cn

本章内容

- 分支限界法基本思想：队列式、优先队列式
- 实际问题：单源最短路径问题、装载问题、布线问题、0-1背包问题、最大团问题、旅行售货员问题、电路板排列问题、批处理作业调度问题

分支限界法 vs. 回溯法

- 求解目标：
  - 回溯法：找出解空间树中满足约束条件的所有解
  - 分支限界法：找出满足约束条件的一个解，或是在满足约束条件的解中找出在某种意义下的最优解
- 搜索方式：
  - 回溯法：深度优先的方式搜索解空间树
  - 分支限界法：广度优先或最小耗费优先的方式搜索解空间树

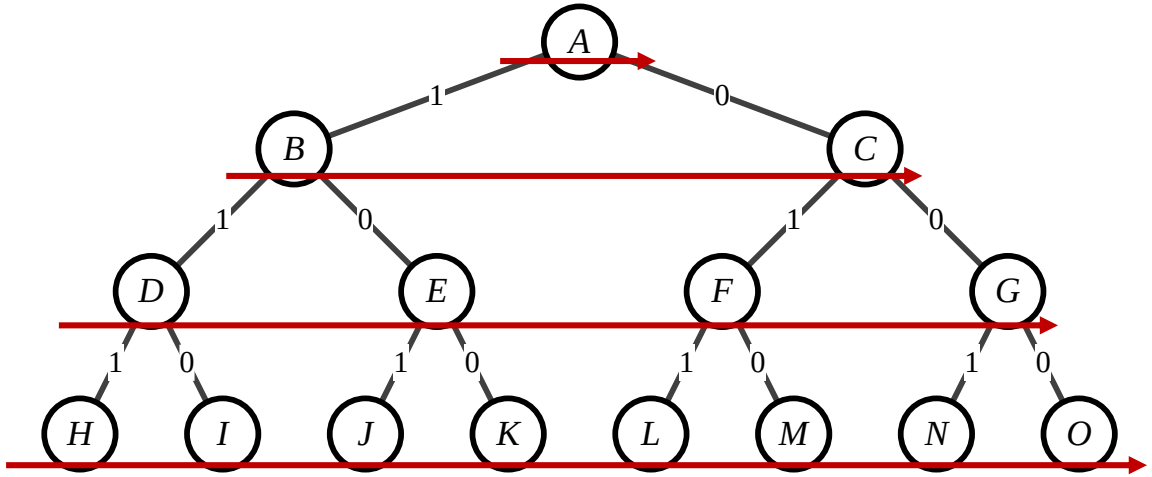
分支限界法基本思想

- 分支限界法常以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树
- 在分支限界法中，每一个活结点只有一次机会成为扩展结点。活结点一旦成为扩展结点，就一次性产生其所有子结点。在这些子结点中，导致不可行解或导致非最优解的子结点被舍弃，其余子结点被加入活结点表中
- 此后，从活结点表中取下一结点成为当前扩展结点，并重复上述结点扩展过程。这个过程一直持续到找到所需的解或活结点表为空时为止

分支限界法基本思想 (续)

■常见的两种分支限界法

- ➢队列式（FIFO）分支限界法：按照队列先进先出（FIFO）原则选取下一个结点为扩展结点
- ➢优先队列式分支限界法：按照优先队列中规定的优先级选取优先级最高的结点成为当前扩展结点



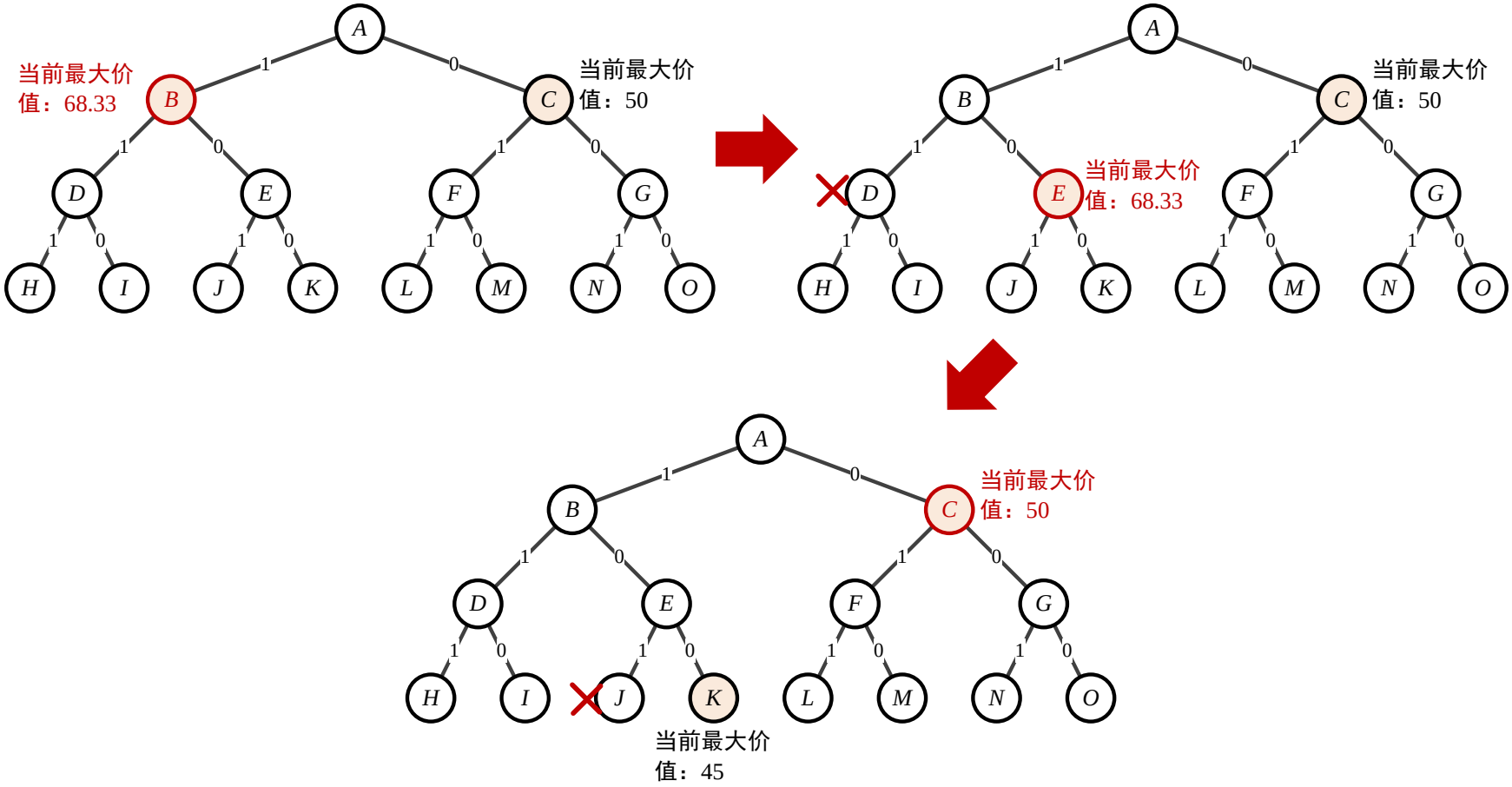
分支限界法求解举例——0-1背包问题

■算法思想：

- 将各物品依其单位重量价值从大到小进行排列
- 优先队列分支限界法：结点的优先级：已装包的物品价值 + 剩下的最大单位重量价值的物品装满剩余容量的价值和
- 首先检查当前扩展结点的左子结点的可行性
- 如果该左子结点是可行结点，则将它加入到子集树和活结点优先队列中
- 当前扩展结点的右子结点一定是可行结点，仅当右子结点满足上界约束时才将它加入子集树和活结点优先队列
- 当扩展到叶结点时为问题的最优值

分支限界法求解举例——0-1背包问题(续)

■例：w=[16, 15, 15], v=[45, 25, 25], W=30, v/w≈[2.8125, 1.67, 1.67]



分支限界法求解举例——0-1背包问题(续)

■例：w=[16, 15, 15], v=[45, 25, 25], W=30, v/w≈[2.8125, 1.67, 1.67]